X-MAS: SUPPORTING THE TEDIOUS WORK OF VALIDATION IN AGENT-BASED SIMULATION

Y.I.L. SUEMATSU* and K. SHIMOHARA, ATR International, Kyoto, Japan,
and Kyoto University, Kyoto, Japan
K. TAKADAMA, ATR International, Kyoto, Japan,
and Tokyo Institute of Technology, Tokyo, Japan
O. KATAI, Kyoto University, Kyoto, Japan

ABSTRACT

Validation is an essential issue in the growing field of agent-based simulation (ABS), as ABS has become a prominent paradigm in the study of social complex systems. However, the main difficulty faced in this validation process is the lack of techniques and tools to assure the reliability of models. Thus, the validation of models is still a tedious task. In our previous work, we proposed *cross-element validation*. This process consists of performing the validation *within* a model by comparing the simulation results of the model under several instances of some of its composite elements. Elements are, for instance, learning mechanisms or network iteration topologies. Even though it is relatively simple, this validation process requires performing several simulations of the model under the possible combinations that exist among a certain number of instances of some elements. In other words, it requires several implementations of the model to account for the above-mentioned combinations of elements. Therefore, tools to support this validation process are required. To support the cross-element validation process for ABS models, this paper presents cross-element validation for multi-agent-based simulation (X-MAS). This tool provides facilities for performing easy cross-element validation of ABS models and also facilitates the implementation of general-purpose ABS models. To illustrate the potential of X-MAS, a cross-element validation of a bargaining game model was performed by evaluating several learning mechanisms applied to the agents. The findings showed that simulation results can be strongly affected by even small variations in the elements. Therefore, cross-element validation should be performed before deep analysis of the implemented model.

Keywords: X-MAS, cross-element validation, agent-based modeling, bargaining

INTRODUCTION

Although agent-based simulation (ABS) is becoming a prominent paradigm in the study of complex social sciences, it still lacks the validation techniques and tools to assure the reliability of the models. One available technique is the so-called docking or alignment of computational models proposed by Axtell et al. (1996). This is a validation process where two models that deal with the same phenomenon are compared, in order to determine whether the results from one computational model match the results of another model. This process requires the replication of one model based on the other model's framework. However, the following

* *Corresponding author address*: Yutaka Inti Leon Suematsu, ATR International, Network Informatics Laboratories, 2-2-2 Hikaridai, Keihanna Science City, Kyoto 619-0288, Japan; e-mail: yutaka@atr.jp.

difficulties make it rather awkward: (1) models may be developed for different purposes, (2) common parts between models may be small in number, and (3) fair evaluation criteria may be difficult to define in order to guarantee the seeming equivalence of the models' simulation results.

In order to overcome the above-mentioned difficulties, our previous work proposed the concept of *cross-element validation* as a process that performs the validation *within* a model (Takadama et al. 2003), as opposed to docking, which consists of validation *between* models. Cross-element validation consists of detecting, analyzing, and comparing the model's macro-behavior under different variations of some of its composite elements. Elements are, for instance, how individual agents store knowledge, how to perform learning, or what iteration network topology is used. By understanding whether and how the elements' implementations affect the model's overall behavior, the reliability of the ABS model is expected to increase.

Cross-element validation must perform several simulations of one model with variations of the composite elements. The number of simulations exponentially increases as both the number of replacements of a certain element and the number of elements to be evaluated increase. Thus, the efficient performance of each simulation is required. Another difficulty is the modification of the model implementations, which is cumbersome. Because of these facts, even relatively simple cross-element validation process is a tedious task. Therefore, tools with the following three requirements are indispensable: (1) easy model implementation, (2) flexibility to simplify model element exchanges, and (3) the construction of efficient programs to accelerate the simulations.

Several ABS libraries and tools are available in the community for facilitating model implementation, such as Swarm (Swarm Development Group 2005), Repast (Repast 2005), and MASON (George Mason University 2005). Unfortunately, their use in the cross-element validation of models is quite difficult because they do not fulfill the above-mentioned requirements.

To support the cross-element validation process of ABS models, this paper presents cross-element validation for multi-agent-based simulation (X-MAS). This tool provides facilities for performing easy cross-element validation of ABS models by providing a framework wherein several variations of some elements involved in the model can be evaluated easily without reimplementing the model for the numerous combinations between elements. X-MAS has been developed as a general-purpose library to support diverse ABS models, ranging from social science to engineering.

To illustrate the potential of X-MAS, the cross-element validation of a bargaining game model was performed, comparing the results of the model with different learning mechanisms, such as evolution strategy (ES), learning classifier system (LCS), and reinforcement learning (RL), and also variations in discrete and continuous values in the representation of the knowledge.

This paper is organized as follows. The next section describes X-MAS and its features and gives a brief description of the composed layers. Then the bargaining model is described, including the simulation results and a brief discussion related to the experimental findings. Finally, a summary and future work are presented.

X-MAS

Features

The cross-element validation for X-MAS provides a rich framework that facilitates both the implementation and cross-element validation of ABS models. Additionally, X-MAS facilitates multi-intelligent agent implementation by providing a framework that embeds the commonly used learning mechanisms and knowledge representation schemes of agents in ABS models. X-MAS is a collection of generic object-oriented programming libraries. Because of performance, scalability, and portability issues, it was developed in Standard C++ with extensive use of STL and BOOST Libraries (2005). Cutting-edge techniques of meta-programming in C++ were employed to provide facilities for easily exchanging elements for their substitutes in the model. These characteristics make it possible to satisfy the flexibility required for the cross-element validation process that was enumerated in the previous section. X-MAS is portable, and the models can be compiled in several operating systems with no modifications. X-MAS only requires the use of compilers that support major features of the C++ Standard, including *partial template specialization*,[1] which is supported in many modern compilers.

X-MAS is considered a general-purpose library for multiagent systems. The core libraries are highly customizable, providing the scalability to build domain-specific libraries and user-customizable libraries. As shown in Figure 1, the simulation is developed in two levels. The left side represents the implementation of the model that will be executed in the back end. Then, visualization of the simulation results are shown and analyzed in an independent process, as shown in the right side of Figure 1. This structure will accelerate the simulation process.

Moreover, X-MAS facilitates the maintenance of the model implementation by keeping the core implementation of the model in one program regardless of the possible variations of the element to be evaluated.

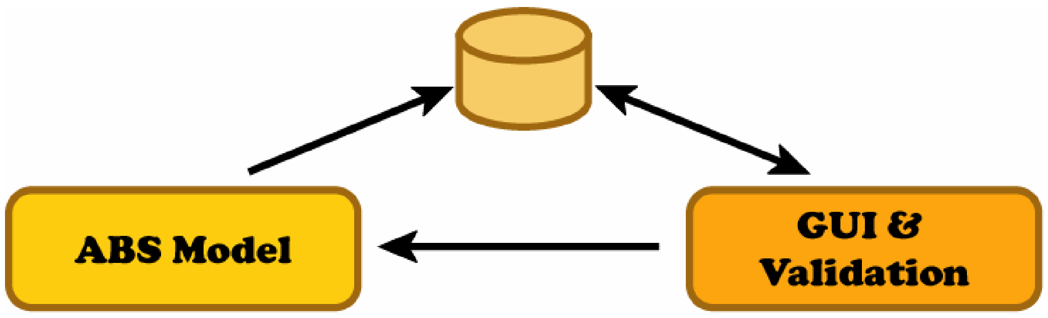

FIGURE 1 Simulation under X-MAS (The left side represents the implementation of the model that is executed in the back end. Then the simulation results are visualized and analyzed by another process, as shown in the right side. Additionally, some GUI interfaces for controlling the simulation and the exchange of elements for performing cross-element validation are available.)

[1] Most modern compilers support the major features of the C++ standard. However, even today, some compilers still do not support important minor features such as partial template specialization. X-MAS makes extensive use of the power of templates to fulfill the requirements of cross-element validation.

Modules

X-MAS consists of three main modules or libraries, as shown in Figure 2: (1) X-MAS core libraries, (2) domain-specific libraries, and (3) visualization and cross-element validation interfaces.

X-MAS Core Libraries

X-MAS provides a set of generic libraries and utilities for easy implementation of simulation models. Several algorithms commonly used in any model implementation are provided with a highly customizability. Some implementations, for instance, consist of the scheduling of agent interaction, selection algorithm, and simulation cycle control.

Domain-specific Libraries

The X-MAS framework allows the implementation of domain-specific libraries by customizing some of the already available libraries. X-MAS provides libraries for supporting intelligent agents by providing several learning mechanism algorithms such as RL, LCSs, genetic algorithms (GAs), and ESs. It also provides genotype-based, rule-based, and other algorithms with discrete and continuous values as knowledge representation schemes. Each algorithm provides a default setting and a highly customizable framework to include variations of the algorithm by allowing parts of the algorithm to be replaced with user's algorithms, such as different crossover algorithms applied to GAs.

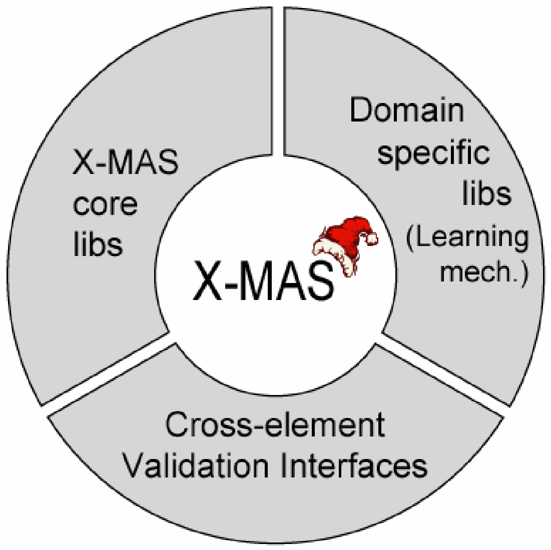
X-MAS core libs
Domain specific libs (Learning mech.)
X-MAS
Cross-element Validation Interfaces

FIGURE 2 The three X-MAS modules

Visualization and Cross-element Validation Interfaces

The main purpose of X-MAS is to provide a set of interfaces to interact with the model. It includes some interfaces to visualize the simulation results of the model under all of the possible combinations between elements. For instance, as represented in Figure 3, it is assumed that the model consists of agents that have learning capabilities and representation capabilities for the knowledge. Three implementations for each involved element are considered. A1 to A3 and B1 to B3 represent implementations of the possible learning mechanisms and knowledge representation schemes, respectively. Therefore, this may require evaluating the model with the nine possible combinations between both elements. As shown in the left side of Figure 3, X-MAS allows the implementation of one model where the combination of the involved element to be evaluated can be easily selected, as opposed to the nine reimplementations of the model with traditional tools. Furthermore, X-MAS provides interfaces to plot the simulation results of the model under all possible combinations among elements, as presented in the right side of Figure 3. Finally, from these results, the equivalence and implications of the implementations of the model elements can be easily evaluated. In the right side of Figure 3, the red box represents equivalent simulation results in six implementations of the model. Therefore, it can be said that the model produces invalid (strange) results when using B3 as element B, as opposed to valid results for the other implementations. As a consequence, the model can be minimally validated under implementations B1 and B2 as element B and all three implementations of element A.

This module is still undergoing development. Because of the fact that there are no standard graphic libraries in the C++ standard, which is a major disadvantage, some interfaces with Python and Java are planned for further development.

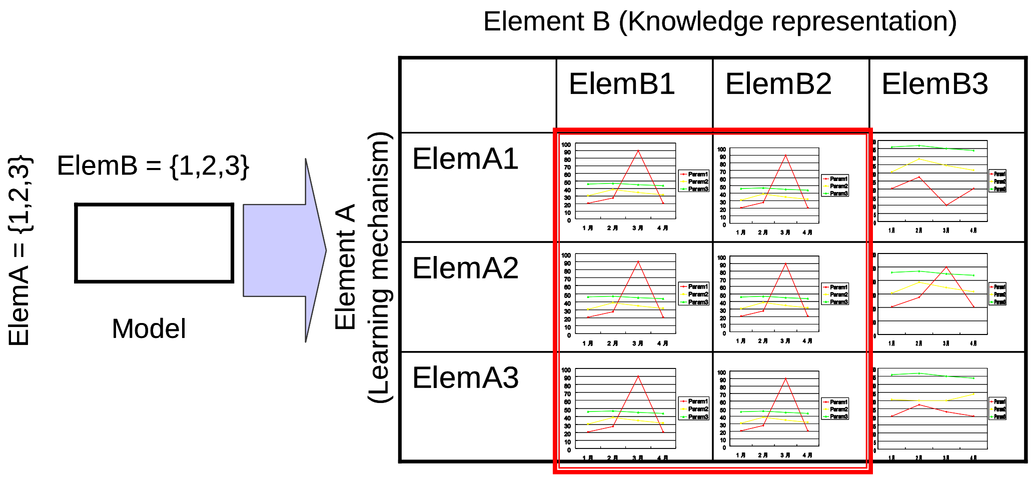

FIGURE 3 Image of the use of X-MAS

CASE STUDY

Bargaining Game

As a concrete example, the bargaining problem (Muthoo 2000) was employed, which addresses a situation where two or more players try to reach a mutually beneficial agreement in order to maximize their profits through negotiations. This problem was selected because it has been studied in the context of game theory (Osborne and Rubinstein 1994) for several years, and its results are well known. Therefore, simulation results can be evaluated by comparing them with the rational behavior of players.

The problem considered in this research is the one proposed by Rubinstein (1982). This model uses the following scenario. Two players, P_1 and P_2, have to reach an agreement on the division of a *pie*. For this purpose, they alternate offers, describing the possible division upon which they would like to settle. The player who receives the offer has to decide whether to accept it or not. If the offer is accepted, the negotiation process ends, and each player receives the share of the pie determined by the concluded contract (e.g., P_1 receives x and P_2 receives $1 - x$ at time t, where x is a value in the interval [0,1]). Otherwise, the receiving player makes a counter-offer, and all of the above steps are repeated until an agreement is reached, or the process is aborted when the limit number of offers is reached; in that case, both players receive a null payoff.

For experimentation, a finite-horizon model was employed, where the maximum number of steps in the game is fixed and known by both players as common information. In the case where the maximum number of steps is one (also known as the *ultimatum game*), the proposer (player P_1) makes the only offer, and the responder player (P_2) can either accept it or not. If P_2 refuses the offer, both players receive a null payoff. Since a rational player always takes actions that maximize her payoff, P_1 tries to keep most of the pie to herself by offering only a minimum share to P_2. Since there are no further steps to be played in the game, P_2 inevitably accepts the tiny offer, under the notion of "anything is better than nothing."

By applying a backward induction reasoning to the situation above, it is possible to analyze situations where the maximum number of steps is greater than one. For the same reason as that of the *ultimatum game*, the player who can make the last offer in a finite game where payoffs are not discounted by time has a great advantage to obtain the larger share[2] of the pie by making a minimum offer (Stahl 1972).

Model

The implemented model was designed in the framework of the bargaining game as follows:

---

[2] In this paper, the terms *payoff* and *agent* are used instead of the terms *share* and *players* for their more general meaning in the bargaining game.

Model Structure

The basic structure of the agents was implemented using the following components as shown in Figure 4. Note that each agent has the same architecture.

< Memory >

Strategies memory. This stores a set of strategies that agents use during negotiation (in Figure 4, the number of strategies is n). Each strategy consists of a fixed number of paired offer (O)/threshold (T) values and the worth of the strategies (w). These strategies are similar to those used in Oliver (1996). The offer and threshold values are encoded by floating point numbers in the interval [0, 1], while the worth values are calculated as averages of acquired payoffs. In this model, agents independently store different strategies, which are initially generated at random.

Selected strategy memory. This stores the strategy selected to confront the strategy of an opponent agent. Figure 4 shows the situation where agent A_1 selects the xth strategy, while agent A_2 selects the yth strategy.

< Mechanism >

Learning mechanism. This modifies both offer and threshold values in order to generate good strategies that acquire a large payoff.

As a concrete negotiation process, agents proceed as follows. Defining $\{O,T\}_i^{A_{\{0,2\}}}$ as the ith offer or threshold value of agent A_1 or A_2, agent A_1 starts proposing the first offer $O_1^{A_1}$. Here, it is counted as one *step* when either agent makes an offer. Then A_2 accepts the offer if $O_1^{A_1} \geq T_1^{A_2}$; otherwise, it makes a counter-offer $O_2^{A_2}$ (i.e., the offer of A_2). This cycle is repeated

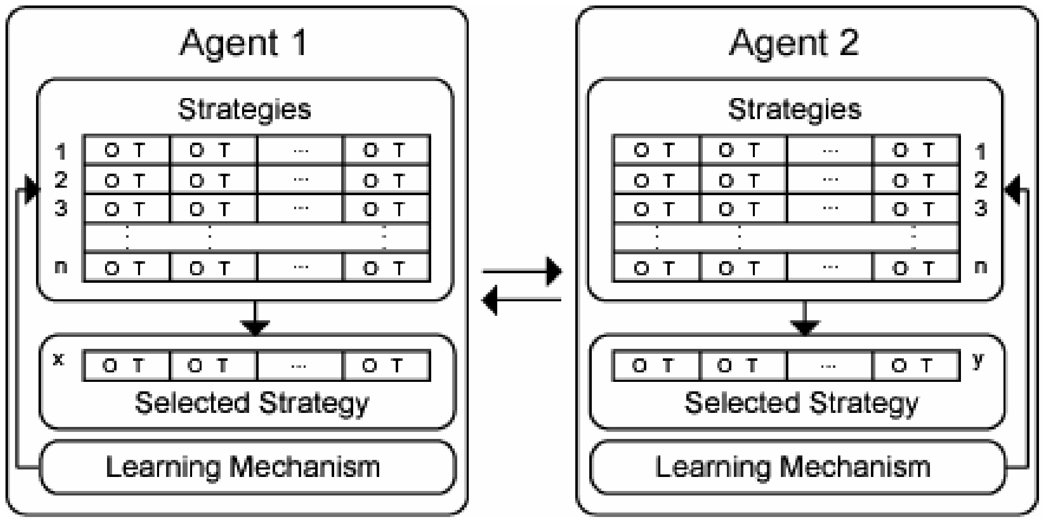

FIGURE 4 Bargaining model structure (Each agent consists of a memory, which is a set of strategies the agent uses during negotiation, and a learning mechanism to improve the strategies.)

until either agent accepts the offer of the other agent or the maximum number of steps is exceeded. To understand this situation, let's consider a simple example where the maximum number of steps is 10, as shown in Figure 5. Following this example, A_1 starts by offering 0.01 to A_2. However, A_2 cannot accept the first offer because it does not satisfy the inequality $O_1^{A_1}(0.01) \geq T_1^{A_2}(0.99)$. Then, A_2 counter-offers 0.01 to A_1. Since A_1 cannot accept the second offer from A_2 for the same reason, this process is repeated until A_1 accepts the 10th offer from A_2 where the offer satisfies the inequality $O_{10}^{A_2}(0.01) \geq T_{10}^{A_1}(0.01)$. In case the negotiation fails, which means that the maximum number of steps has been exceeded, both agents can no longer receive any payoff (i.e., they receive 0 payoff). Here, this is counted as one *confrontation* when the above negotiation process ends (satisfactory or unsatisfactory.)

Furthermore, the worth of each strategy is calculated by the average of payoffs acquired during a fixed number of confrontations (*CONFRONTATION*), where the strategies of the other agent are randomly selected in each confrontation. For example, the *x*th strategy of A_1 in Figure 4 confronts the randomly selected strategies of the other agent in the predefined number of confrontations, and then the worth of the *x*th strategy is calculated by the average of payoffs acquired during these confrontations. Since each agent has *n* number of strategies, the $(CONFRONTATION \times n \times 2)$ number of confrontations is required to calculate the worth of all strategies of both agents. Here, it is counted as one *iteration* when the worth of all strategies of both agents is calculated.

Elements for Cross-element Validation

The focus of this case study is to make some comparative studies to investigate the influence of different learning mechanisms and knowledge representation schemes (Takadama et al. 2003). For this purpose, each element was designed as follows.

Learning Mechanisms

When the learning mechanisms of agents are being implemented, several mechanisms can be considered. Among the many useful learning mechanisms, the following were employed: (1) ES (Back et al. 1991, 1993), (2) LCS (Goldberg 1989; Holland et al. 1986), and (3) RL (Sutton and Barton 1998).

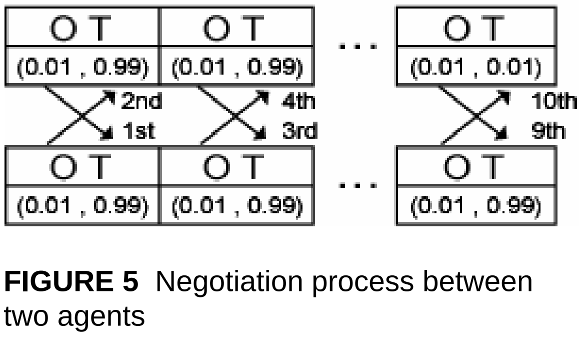

FIGURE 5 Negotiation process between two agents

Knowledge Representation Schemes

In the bargaining game, the representation of the agents' strategies must be considered, though there are no standard guidelines. From this fact, the following two types of knowledge representation capabilities were employed: (1) continuous real numbers (e.g., 0.01…) and (2) real numbers restricted to two decimal digits (e.g., 0.01; called discrete numbers in this paper). The reason why this knowledge representation was employed is because (1) social scientists may take the latter case for a concise representation and (2) a real number in offer and threshold values is critical in the bargaining game.

Simulation Results

Figure 6 shows the simulation results of the possible combinations among learning mechanisms (ES, LCS, and RL) with the two knowledge representation schemes (continuous and discrete). All figures indicate the results of the payoff. The vertical axis indicates the payoff, while the horizontal axis indicates the iteration number. In particular, Figure 6 shows the payoff of agent A1 in the lower lines and that of A2 in the upper lines. Figures 6(a) and 6(b) represent the results when using continuous values as knowledge representation schemes with ES and LCS, respectively, as learning mechanisms. Similarly, Figures 6(c), 6(d), and 6(e) represent the results when using discrete values as knowledge representation schemes with ES, LCS, and RL, respectively, as learning mechanisms. These results show that simulation results do not exhibit the same tendency when different learning mechanisms or knowledge representation schemes are applied to agents.

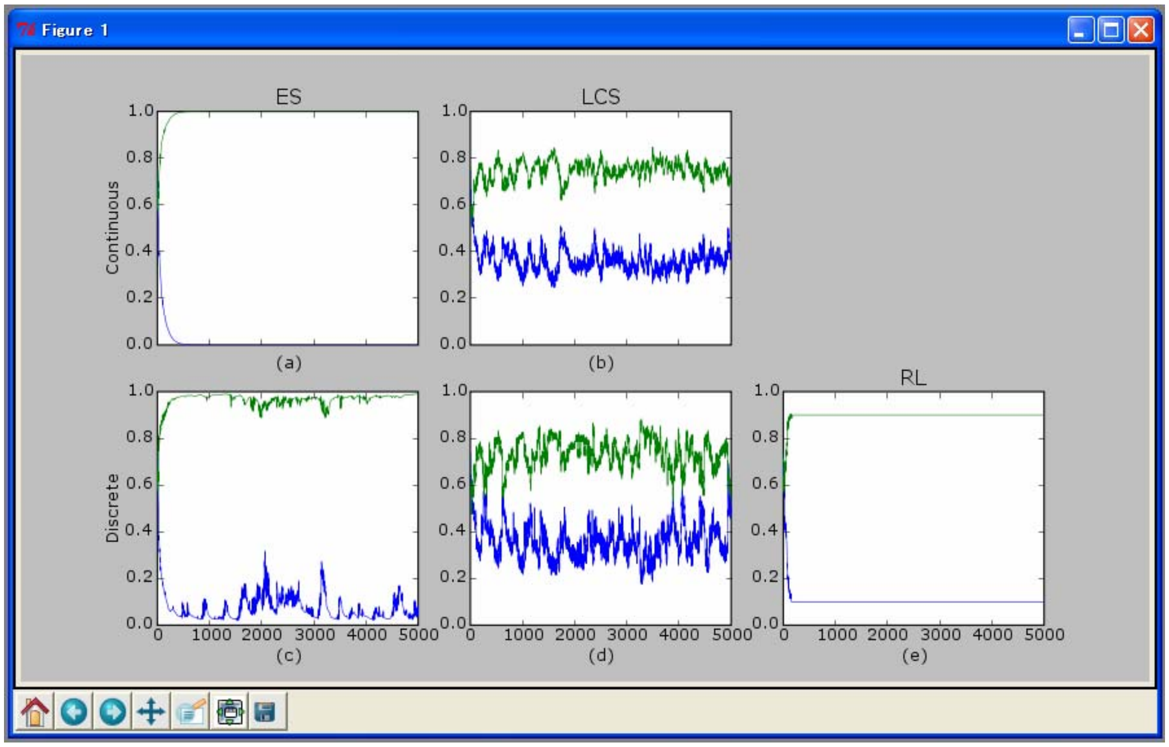

FIGURE 6 Simulation results

Discussion

Results of Cross-element Validation

Theoretical results from game theory prove that both rational agents A_1 and A_2 receive the minimum and maximum payoffs at the final negotiation process, respectively. This is because A_1 in our simulations has to accept any small offer proposed by A_2 at the 10th negotiation process; otherwise, A_1 cannot receive any payoff (i.e., it receives a null payoff). Therefore, it was expected that learning agents can acquire the maximum and minimum payoffs.

Analyzing Figures 6(a) and 6(b) shows that the payoff of ES-based agents finally converges at the mostly maximum or minimum value (i.e., 1 or 0), while that of LCS-based agents neither converges at a certain value nor becomes close to the maximum or minimum value. From Figures 6(a) and 6(c), it is observed that the results of ES-based agents using discrete knowledge representation degrade the results obtained when using continuous knowledge representation (note the rather wavy lines in Figure 6(c)). Finally, from Figures 6(c) and 6(e), it is observed that the payoff of ES-based agents was effected using two decimal digits, while RL-based agents converge at the mostly maximum or minimum value (i.e., 0.9 or 0.1).

These results show that ES-based agents with continuous knowledge representations and RL-based agents with discrete knowledge representations could produce results as expected by game theory. Therefore, both models are minimally validated.

From this analysis, it can be concluded that simulation results are sensitive to the learning mechanisms applied to agents. Also, even minor considerations in the knowledge representation, particularly discrete and continuous representations, may produce unexpected results.

As a result, it is strongly recommended that some cross-element validation of models be performed before deep analysis and interpretation of their simulation results.

X-MAS Compared with Other Tools

In order to help researchers in the field of social sciences simulate their models, several tools have been developed to reduce the difficulties of the programming process and enhance the understanding of the outcomes (e.g., Repast, Swarm, and Mason). However, performing cross-element validation will require the knowledge of some internal libraries to easily exchange elements in the model. In several cases, it may require reimplementation for all possible substitute elements in the model. The reason for this is that they are not designed for validation purposes but for easy program implementation. X-MAS, on the other hand, was designed to support the cross-element validation of ABS models and to facilitate program implementation.

X-MAS provides a framework for implementing generic models, and several variations of the model can be performed more easily. It is expected to be considered as a framework for the replication of models.

SUMMARY

Although ABS is becoming an essential tool in the study of complex social sciences, the validation of ABS models is still an important issue to be considered. Cross-element validation was proposed in our previous work. This process consists of performing the validation within a model by comparing the simulation results of the model under several instances of some of its composite elements. To support the cross-element validation process of ABS models, this paper presented the cross-element validation for multi-agent-based simulation (X-MAS). This tool provides facilities for simplifying the cross-element validation of ABS models. It also facilitates the implementation of general-purpose ABS models. The potential of X-MAS was tested by means of a bargaining game model, by evaluating several learning mechanisms applied to the agents. It showed that simulation results can be strongly affected by even small variations in the elements. In particular, arbitrary assumptions in the learning mechanism and knowledge representation schemes may produce unexpected results. Therefore, cross-element validation should be performed before deep analysis and interpretation of the implemented model.

Further research includes (1) implementating several GUIs for interaction with models and (2) performing cross-element validation of several models.

ACKNOWLEDGMENTS

This research was conducted as part of “Research on Human Communication” with funding from the National Institute of Information and Communications Technology (NICT), Japan. The research was also supported in part by the Sasagawa Scientific Research Grant from the Japan Science Society. In addition, it was supported in part by a Grant-in-Aid for Scientific Research (Young Scientists (B), 17700139) from the Ministry of Education, Culture, Sports, Science and Technology (MEXT), Japan. We would like to express our thanks for all of the above support.

REFERENCES

Axtell, R., R. Axelrod, J.M. Epstein, and M. Cohen, 1996, “Aligning simulation models: A case study and results,” *Computational and Mathematical Organization Theory (CMOT) 1*(1):123–141.

Back, T., F. Hoffmeister, and H. Schwefel, 1991, “A survey of evolution strategies,” in *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 2–9.

Back, T., F. Hoffmeister, and H. Schwefel, 1993, “Evolutionary programming and evolution strategies: Similarities and differences,” in *Proceedings of the 2nd Annual Conference on Evolutionary Programming*, pp. 11–22.

Boost Libraries, 2005, home page; available at http://www.boost.org/.

George Mason University, 2005, MASON; available at http://cs.gmu.edu/~eclab/projects/mason/.

Goldberg, D., 1989, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley.

Holland, J., K. Holyoak, R. Nisbett, and P. Thagard, 1986, *Induction*, The MIT Press.

Muthoo, A., 2000, "A Non-technical Introduction to Bargaining Theory," *World Economics 1*(2):145–166.

Oliver, J., 1996, *On Artificial Agents for Negotiation in Electronic Commerce*, Ph.D. thesis, University of Pennsylvania.

Repast, 2005, home page; available at http://repast.sourceforge.net/.

Osborne, M., and A. Rubinstein, 1994, *A Course in Game Theory*, The MIT Press.

Rubinstein, A., 1982, "Perfect Equilibrium in a Bargaining Model," *Econometrica 50*(1):97–109.

Stahl, I., 1972, *Bargaining Theory*, Economics Research Institute at the Stockholm School of Economics.

Sutton, R., and A. Barto, 1998, *Reinforcement Learning: An Introduction*, The MIT Press.

Swarm Development Group, 2005, home page; available at http://www.swarm.org.

Takadama, K., Y. Leon, N. Sugimoto, E. Nawa, and K. Shimohara, 2003, "Cross-element validation in multiagent-based simulation: Switching learning mechanisms in agents," *Journal of Artificial Societies and Social Simulation 6*(4).